# Presentation Transcript: Sentinel AI Sentiment Dashboard

## Part 1: Jupyter Notebook Code Walkthrough (The Model Training)

**(Start your screen share on the Jupyter Notebook)**

### 1. Introduction and Setup (Cells 1-3)

"Good morning/afternoon, everyone. Before we dive into the live application, I want to quickly walk you through the core engine powering the **Sentinel AI Sentiment Dashboard**—our deep learning model, built and trained right here in this Jupyter Notebook."

"The first few cells handle our setup. We import essential libraries: **Pandas** for data handling, **TensorFlow/Keras** for building and training our neural network, and **Scikit-learn** for utilities like splitting our data."

"We define some initial constants, like the maximum length of our chat messages and the size of our vocabulary, which ensures consistency when we pass data from this training environment to the live application."

### 2. Data Loading and Exploration (Cells 4-6)

"Next, we load our labeled training data. This is a crucial step, as the quality of the data directly determines the model's accuracy. Our dataset consists of thousands of customer and agent chat messages, each labeled with a sentiment score—from negative, like **-1.0**, to positive, like **+1.0**."

"We perform a quick data exploration to check for balance. It’s important to see if we have roughly equal representations of positive, neutral, and negative examples. If one sentiment is dominant, the model might become biased, so we'd know to address that here."

### 3. Preprocessing and Tokenization (Cells 7-9)

"Machine learning models don't understand text directly; they understand numbers. This is where **preprocessing** comes in."

"First, we define a **Tokenizer**. Think of the tokenizer as a vocabulary builder. It scans all the text and assigns a unique integer ID to every single word. This converts our raw sentences into sequences of numbers."

"We then use **Padding**. Since chat messages vary in length—some are short, some are long—we must standardize them to a fixed size. Padding adds zeros to the end of shorter sequences, ensuring every input array has the exact same dimensions for the neural network."

### 4. Model Definition (Cells 10-12)

"Now, we define the architecture of our deep learning model. We've chosen a specialized type of neural network: a **Recurrent Neural Network (RNN)**, specifically using **Long Short-Term Memory (LSTM)** layers."

"The structure is straightforward:"

* "**Embedding Layer:** This is the first layer. It takes those simple integer IDs and converts them into dense, multi-dimensional vectors. This helps the model understand the contextual meaning of words."
* "**LSTM Layer:** This is the magic. LSTM is designed to process sequences, remembering information from earlier words and forgetting irrelevant ones. This makes it highly effective for understanding the context of a message."
* "**Dense Layers:** These are standard layers that receive the output from the LSTM and collapse the context down into a single prediction—our final sentiment score."
* "The final output layer uses a **linear activation** to give us a continuous sentiment score between our -1.0 and +1.0 targets."

### 5. Training and Evaluation (Cells 13-15)

"With the architecture defined, we compile the model, specifying the **optimizer** and the **loss function**. We use **Mean Squared Error (MSE)** as our loss, as it's excellent for regression tasks where we are predicting a continuous numerical value."

"We then initiate the training loop, feeding the model our prepared data over several **epochs**."

"Finally, we run the model on our **validation set**—data the model has never seen before. The resulting metrics, like the low Mean Absolute Error, confirm that the model has generalized well and is highly accurate at predicting sentiment on new, unseen chat messages."

### 6. Saving the Model (Cell 16)

"The last step is critical: we save the trained model weights and configuration. This **saved model file** is what we deploy to our Streamlit application, allowing the dashboard to utilize this complex deep learning logic instantly, without needing to re-train."

## Part 2: Sentinel AI Application Demo Transcript

**(Transition your screen share to the Sentinel AI Streamlit application)**

### 7. Application Overview and Value Proposition

"Welcome to the **Sentinel AI Chat Sentiment Dashboard**. Our goal was to take that powerful model we just saw and wrap it in a simple, highly valuable tool for contact center managers and agents."

"In a high-volume chat environment, managers can't read every message. This application acts as an **early warning system**, providing a real-time, objective pulse on every customer interaction."

### 8. The Live Chat Demonstration

"I'm going to run a quick conversation simulation here using the sidebar dropdown, let's select **'Simulation: Negative Escalation'**."

"Notice how the chat log populates. As the messages appear, the application instantly processes them through the deployed AI model."

**(Click to show a customer message with negative sentiment)**

"See this message? 'I have been waiting for 30 minutes and this is ridiculous!' Immediately, the model flags this with a **red background** and a score of **-0.85**. This color-coded scoring is the manager's signal to check on this agent or conversation, or for the agent to recognize the urgency."

**(Click to show a positive resolution message)**

"Now, as the agent successfully de-escalates and offers a clear solution, the score of the agent's response, and the customer's final reply, shift to **green**, maybe a **+0.60**. This confirms a successful recovery."

### 9. The Dashboard: Real-Time Metrics and Trends

"On the right, we have our real-time dashboard. This updates instantly with every new message."

* "**Overall Sentiment:** Right now, the overall score is **-0.15**. This gives you an average health score for the entire transcript."
* "**Per-Message Trend Chart:** This line graph is the most useful visualization. You can clearly see the dramatic dip when the customer first escalated, and then the slow, steady rise back towards zero as the agent stabilized the situation. It visually proves the agent’s ability to recover a negative interaction."
* "**Cumulative Trend Chart:** This shows the net change. It quickly tells you if the conversation is trending positively or if it's spiraling downwards."

### 10. Batch Analysis and Model Adaptation (Advanced Features)

"Finally, let's look at the power features that future-proof this tool."

"First, the **Batch Analysis** section. You don't have to monitor just the live feed. You can upload a **CSV file** containing thousands of conversations from last month. The app processes them overnight and provides aggregate reports: which agents have the lowest overall sentiment scores, and what are the top five phrases associated with high customer frustration?"

"Second, the **Model Configuration and Training tab**."

"This is key for evolving business needs. If Sentinel Bank starts using new industry jargon, the sentiment model needs to learn it. Here, a manager can easily upload a new, small set of labeled training data and click **'Re-Train Model'**."

"The application safely re-trains the deployed model on the backend using the new data, without needing data scientist intervention. This ensures our AI engine is always adapting to the bank’s latest customer interaction patterns, keeping the sentiment analysis highly accurate."

### 11. Conclusion

"The Sentinel AI Dashboard is more than just a chat application; it's a **proactive intelligence tool**. It allows us to intervene in moments of crisis, reinforce positive agent behavior, and continuously improve our service delivery, all powered by an integrated, adaptable deep learning model."

"Thank you. I'm now open for any questions you might have about either the underlying code or the application's functionality."